Challenge 3

CY6740 – Machine Learning in Cyber-Security

Name: Tejas Krishna Reddy

NUID: 001423166

Solution:

We have a dataset with 37,438 instances. In this dataset there are 13 different features and a label. The label is of 2 different classes, namely, "human" and "bot". This makes the current problem a binary classification problem. Among the 13 features, we have 6 categorical features and 7 numerical features.

Below is a pseudo algorithm to achieve the classification purpose as asked:

1. First read the CSV into a pandas dataframe structure for easier manipulations.
2. Remove all the instances that have missing values (NaN's). After removing all the missing instances, we remain with a dataset of 29479 instances.
3. Now, we have a feature 'location', in which there are 'unknown' present. Hence, we remove all the instances where the location is 'unknown'. After doing this we remain with a dataset of 21192 instances.
4. Next, we do label encoding for all the 6 categorical features – 'default_profile', 'default_profile_image', 'geo_enabled', 'lang', 'location', 'verified'.
5. Now, we encode the labels. Class Human – 1, Bot – 0
6. Now, divide the dataset into features (X) and labels (y).
7. Scale all the features using sklearn standardScaler module.
8. Divide the dataset into training dataset (80%) and testing dataset (20%) which leaves us with 16953 training instances and 4239 testing instances.
9. Now, import SVM and KNN classifiers from sklearn module.
10. Tune the hyper-parameters to get the best possible accuracies and look out for overfitting or underfitting. Balancing the class weights is extremely important here, else svm would just predict 1 for all the instances.
11. Use 5-fold cross validation on the training data to check on the accuracies over different fold datasets.
12. Save/pickle the model params and check its working on test dataset.

Results:

The average training accuracies for SVM are as follows:

```
scores
```

```
array([0.73341197, 0.72073135, 0.73105279, 0.73687316, 0.73480826])
```

```
print('Mean Accuracy obtained from training dataset using SVM: ', np.mean(scores))
```

```
Mean Accuracy obtained from training dataset using SVM:  0.731375504654434
```

Average Testing Accuracy using SVM: 0.720685660822882

The Average Training Accuracies using KNN Algorithm:

```
scores1
```

```
array([0.79504571, 0.79917428, 0.80094367, 0.80353982, 0.79970501])
```

```
print('Mean Accuracy obtained from training dataset using KNN: ', np.mean(scores1))
```

```
Mean Accuracy obtained from training dataset using KNN:  0.7996817012584936
```

Average Testing Accuracy using KNN: 0.7926368871265954

Confusion Matrix for SVM is as follows:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, clf.predict(X_test))
```

```
array([[ 642,  278],
       [ 900, 2419]], dtype=int64)
```

Confusion Matrix for KNN is as follows:

```
confusion_matrix(y_test, clf2.predict(X_test))
```

```
array([[ 419,  501],
       [ 337, 2982]], dtype=int64)
```

Precision Scores for SVM and KNN are as follows:

```
from sklearn.metrics import precision_score
print('Human Precision SVM - ', precision_score(y_test, clf.predict(X_test)))
print('Human Precision KNN - ', precision_score(y_test, clf2.predict(X_test)))
```

```
Human Precision SVM -  0.8969225064886911
Human Precision KNN -  0.8561584840654608
```

```
## precision in classifying Bots - 0's.
print('Bot Precision SVM - ', 642/920)
print('Bot Precision KNN - ', 419/920)
```

```
Bot Precision SVM -  0.6978260869565217
Bot Precision KNN -  0.45543478260869563
```

Conclusion:

Plainly looking at accuracy scores of both the algorithms it seems like KNN is doing better than SVM. But once we look at the confusion matrices for both the algorithm's we could see that SVM is doing way better than KNN in identifying the bots. Precision scores are better for SVM and hence, I would recommend using SVM for the current problem even though the overall accuracy seems to be better for KNN algorithm.

Challenge 3 - Classifying twitter accounts based on their creation by a bot Vs Human

- **Author: Tejas Krishna Reddy**
- **Date: 10th October 2020**
- **NUID: 001423166**

In [3]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
## Reading the dataset into pandas dataframe structure for easier processin
df = pd.read_csv('Dataset_Challenge3.csv')
```

In [5]:
```
## Display the first 5 columns of dataset
df.head(5)
```

Out[5]:

ers_count	friends_count	geo_enabled	id	lang	location	statuses_count	verifie
1589	4	False	7.870000e+17	en	unknown	11041	Fals
860	880	False	7.960000e+17	en	Estados Unidos	252	Fals
172	594	True	8.760000e+17	en	Los Angeles, CA	1001	Fals
517	633	True	7.560000e+17	en	Birmingham, AL	1324	Fals
753678	116	True	4.647813e+08	en	England, United Kingdom	4202	Tru

Remove instances with missing values

In [6]:
```
## Removing NaN values
df.dropna(inplace = True)
df.shape
```

Out[6]: (29479, 14)

In [7]:
```
### Remove instances that have 'unknown' in them
df = df[df['location'] != 'unknown']
df.shape
```

Out[7]:
```
(21192, 14)
```

In [8]:
```
## Print all the column names
df.columns
```

Out[8]:
```
Index(['default_profile', 'default_profile_image', 'favorites_count',
       'followers_count', 'friends_count', 'geo_enabled', 'id', 'lang',
       'location', 'statuses_count', 'verified', 'average_tweets_per_day',
       'account_age_days', 'account_type'],
      dtype='object')
```

Label Encoding the Categorical Features.

In [9]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

### Transforming each individual feature,one at a time.
df['default_profile'] = le.fit_transform(df['default_profile'])
df['default_profile_image'] = le.fit_transform(df['default_profile_image'])
df['geo_enabled'] = le.fit_transform(df['geo_enabled'])
df['lang'] = le.fit_transform(df['lang'])
df['location'] = le.fit_transform(df['location'])
df['verified'] = le.fit_transform(df['verified'])

## Label encode the predictor variable too.
df['account_type'] = le.fit_transform(df['account_type'])
```

In [10]:
```
## All numerical values, display to check if everything is working as inten
df.head(5)
```

Out[10]:

llowers_count	friends_count	geo_enabled	id	lang	location	statuses_count	verifie
860	880	0	7.960000e+17	9	2657	252	(
172	594	1	8.760000e+17	9	4475	1001	(
517	633	1	7.560000e+17	9	1176	1324	(
753678	116	1	4.647813e+08	9	2628	4202	
27394	542	0	8.010000e+17	1	10196	11513	(

Seperate the Features and Labels

In [11]:
```
target_variable = 'account_type'

X = df.drop([target_variable], 1)
y = df[target_variable]
```

Using Standard Scaling technique to scale all the features.

In [12]:
```
from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit_transform(X)
X = pd.DataFrame(X)
```

Divide the dataset into test (20%) and train (80%).

In [13]:
```
from sklearn import model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, t
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(16953, 13) (16953,)
(4239, 13) (4239,)
```

Applying Kfold(5-Fold) cross-validation on training and testing dataset using SVM Classifier

In [39]:
```
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
clf = SVC(kernel='linear',degree=3, class_weight = 'balanced', shrinking=Tr
scores = cross_val_score(clf, X_train, y_train, cv=5)
scores
```

Out[39]:
```
array([0.73341197, 0.72073135, 0.73105279, 0.73687316, 0.73480826])
```

In [40]:
```
print('Mean Accuracy obtained from training dataset using SVM: ', np.mean(s
```

```
Mean Accuracy obtained from training dataset using SVM:  0.731375504654434
```

In [41]:
```
## Using the same trained models on test dataset
test_scores = cross_val_score(clf, X_test, y_test, cv=5)
test_scores
```

Out[41]:
```
array([0.71933962, 0.73820755, 0.7370283 , 0.70165094, 0.70720189])
```

In [42]:
```
print('Mean Accuracy obtained from testing dataset using SVM: ', np.mean(te
```

```
Mean Accuracy obtained from testing dataset using SVM:  0.720685660822882
```

Applying Kfold(5-Fold) cross-validation on training and testing dataset using KNN Classifier

In [24]:
```
from sklearn.neighbors import KNeighborsClassifier as KNN
```

In [26]:
```
clf2 = KNN(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30
scores1 = cross_val_score(clf2, X_train, y_train, cv=5)
scores1
```

Out[26]:
```
array([0.79504571, 0.79917428, 0.80094367, 0.80353982, 0.79970501])
```

In [27]:
```
print('Mean Accuracy obtained from training dataset using KNN: ', np.mean(s
```

```
Mean Accuracy obtained from training dataset using KNN:  0.7996817012584936
```

In [28]:
```
## Using the same trained models on test dataset
test_scores1 = cross_val_score(clf2, X_test, y_test, cv=5)
test_scores1
```

Out[28]:
```
array([0.78537736, 0.80542453, 0.80188679, 0.79009434, 0.78040142])
```

In [30]:
```
print('Mean Accuracy obtained from testing dataset using SVM: ', np.mean(te
```

```
Mean Accuracy obtained from testing dataset using SVM:  0.7926368871265954
```

Precision and Confusion Matrix for both KNN and SVM

In [43]:
```
clf = SVC(kernel='linear',degree=3, shrinking=True, class_weight = 'balance
clf2 = KNN(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30
```

In [44]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, clf.predict(X_test))
```

Out[44]:
```
array([[ 642,  278],
       [ 900, 2419]], dtype=int64)
```

In [45]:
```
confusion_matrix(y_test, clf2.predict(X_test))
```

Out[45]:
```
array([[ 419,  501],
       [ 337, 2982]], dtype=int64)
```

In [36]:
```
y_test.value_counts()
```

Out[36]:
```
1    3319
0     920
Name: account_type, dtype: int64
```

In [51]:
```
from sklearn.metrics import precision_score
print('Human Precision SVM - ', precision_score(y_test, clf.predict(X_test)
print('Human Precision KNN - ', precision_score(y_test, clf2.predict(X_test
```

```
Human Precision SVM -  0.8969225064886911
Human Precision KNN -  0.8561584840654608
```

In [49]:
```
## precision in classifying Bots - 0's.
print('Bot Precision SVM - ', 642/920)
print('Bot Precision KNN - ', 419/920)
```

```
Bot Precision SVM -  0.6978260869565217
Bot Precision KNN -  0.45543478260869563
```

In [ ]:
```

```